FileMaker® 9

ODBC and JDBC Guide

FileMaker, Inc.
5201 Patrick Henry Drive
Santa Clara, California 95054

For more information, visit our website at www.filemaker.com.

Edition: 01

Contents

Chapter 1
Introduction

This guide explains concepts and details to help you share FileMaker® data with other applications using ODBC (Open Database Connectivity) and JDBC (Java Database Connectivity). This guide also documents how the ODBC and JDBC client drivers, when used with FileMaker Pro and FileMaker Server Advanced, support the industry standards for ODBC, JDBC, and SQL (Structured Query Language).

You can use FileMaker Pro, FileMaker Pro Advanced, or FileMaker Server Advanced to create and test your database solution. You can then share your FileMaker database solution as a data source with ODBC- and JDBC-compliant applications. You must install the FileMaker ODBC and JDBC drivers on the machine where your third-party application is installed.

Important This guide describes using FileMaker software as a data source. For step-by-step information on using FileMaker Pro as an ODBC client application, see FileMaker Pro Help.

The following table gives an overview of how to use ODBC and JDBC with FileMaker software.

What do you want to do?	How do you do it?	See
Use FileMaker Pro as a data source/share FileMaker Pro data	1. SQL queries 2. JDBC	This guide
Use FileMaker Pro as an ODBC client/ access ODBC data	1. Interactively via the relationships graph 2. One-time, static via ODBC import or File menu > Open. Also, the Import Records script step and the Execute SQL script step	FileMaker Pro Help

About this guide

- For information on using ODBC and JDBC with previous versions of FileMaker Pro, see www.filemaker.com/downloads.
- This guide assumes that you are familiar with the basics of using ODBC and JDBC, and constructing SQL queries. Refer to a third-party book for more information on these topics.
- This guide uses “FileMaker Pro” to refer to both FileMaker Pro and FileMaker Pro Advanced, unless describing specific FileMaker Pro Advanced features.

Note You can download PDFs of FileMaker documentation from www.filemaker.com/downloads. Any updates to this document are also available from the website.

Using a FileMaker database as a data source

ODBC and JDBC are application programming interfaces (APIs). These APIs give client applications a common language for interacting with a variety of data sources and database services, including FileMaker Pro and FileMaker Server Advanced.

All applications that support ODBC and JDBC recognize a basic subset of SQL statements. Working with SQL, you can use other applications (like spreadsheets, word processors, and reporting tools) to view, analyze, and modify FileMaker data.

SQL is passed through the ODBC and JDBC interfaces to the FileMaker host of the data source, performing queries such as `SELECT first_name, last_name FROM customers WHERE city='Paris'` and making updates such as the creation of a new record with `INSERT INTO customers (first_name, last_name) VALUES ('Jane','Smith')`.

Accessing a hosted FileMaker Pro database

With either FileMaker Server Advanced or FileMaker Pro, you can host a FileMaker database file as a data source, sharing your data with other applications using ODBC and JDBC. The following table describes what each FileMaker product allows.

This FileMaker product	Allows
FileMaker Server Advanced	Up to 50 connections and supports local access (same computer) and remote access (both for middleware such as web servers, and for remote client access from desktop productivity applications).
FileMaker Pro	Up to five connections and supports local access (same computer) only.

The ODBC and JDBC plug-in components you need for sharing your data with other applications is installed with FileMaker Server Advanced and FileMaker Pro.

To access a hosted FileMaker database file, you need to install the corresponding ODBC or JDBC client driver. Install the client driver on the machine on which the third-party application is installed. See chapter 2, “Installing FileMaker ODBC and JDBC client drivers” for information on installing the driver files needed for accessing a FileMaker data source.

If your FileMaker database solution uses more than one FileMaker database file, all of the database files must be on the same computer.

Chapter 5, “Supported standards,” describes the SQL statements that the ODBC and JDBC client drivers support when used with FileMaker Pro and FileMaker Server Advanced.

Important If you disable ODBC/JDBC sharing after it's already been on, a data source hosted by FileMaker Server Advanced or FileMaker Pro immediately becomes unavailable. The database administrator doesn’t have the capability to alert ODBC and JDBC client applications about the data source’s availability (the administrator can communicate only with FileMaker database file clients). No errors are reported, and the client application should notify users that the data source is not available and transactions cannot be completed. If a client application attempts to connect to an unavailable FileMaker database file, a message explains that the connection failed.

Limitations with third-party tools

- Microsoft Query Wizard: In a FileMaker data source, you cannot access table or column names that use High ASCII or double-byte characters. Instead, use Microsoft Query and manually enter the characters, enclosed in double quotation marks.
- Microsoft Access: In a FileMaker data source, you cannot access table or column names that use High ASCII or double-byte characters.

Networking requirements

You need a TCP/IP network when using FileMaker Server Advanced to host a FileMaker database file as a data source over a network. FileMaker Pro supports local access (same computer) only.

Updating files from previous versions

If you used LDAC (Local Data Access Companion) or RDAC (Remote Data Access Companion) in versions prior to FileMaker Server 5.5 to share a database file, you'll need to make two changes. First, you need to set up users to belong to an account that has the extended privilege of Access via ODBC/JDBC (as part of the account's privilege set).

See FileMaker Pro Help for information about sharing via ODBC/JDBC and setting up accounts and privilege sets.

Installing current drivers

If you installed a driver from versions prior to FileMaker Server 9 Advanced or FileMaker Pro 9, you must uninstall the earlier driver and install the driver for version 9. For more information, see chapter 2, "Installing FileMaker ODBC and JDBC client drivers."

Note You have to create a DSN for each FileMaker database file you want to access as a data source. If you have previously set up access through one DSN that allows tables to be spread among several FileMaker database files, you'll need to consolidate those tables into a single database file (or create several DSNs).

Chapter 2
Installing FileMaker ODBC and JDBC client drivers

These instructions help you install the drivers needed to access a FileMaker data source from third-party and custom applications via ODBC (Open Database Connectivity) and JDBC (Java Database Connectivity). The client drivers are available through a separate installation on your FileMaker CD or electronic download in the folder \xDBC.The latest versions of the client drivers are also available from www.filemaker.com/support/technologies.

If you'll be hosting a FileMaker database file using FileMaker Server Advanced, make the client drivers available to remote users.

After installing the client driver you need, you can configure the driver to access a FileMaker data source and construct SQL (Structured Query Language) queries to interact with the data.

Abiding by the license agreement

The ODBC and JDBC client drivers are the driver portions of the FileMaker software that allow third-party applications or custom applications to access FileMaker files as ODBC or JDBC data sources.

Hardware and software requirements

To install and use the ODBC and JDBC client drivers, you need the following minimum equipment and software:

ODBC client driver requirements (Windows)

- Pentium 300 MHz or faster
- 64 MB total RAM for Windows 2000, 128 MB total RAM for Windows XP
- Microsoft Data Access Components (MDAC) 2.8 SP1
- Windows MDAC 6.0 for Vista

ODBC client driver requirements (Mac OS)

- Apple G3 or faster (no G3 upgrade cards)
- 128 MB total RAM
- Mac OS X version 10.3.9 or 10.4 (the software may also work with later versions certified by FileMaker)

JDBC client driver requirements

Applications using	Require
JDBC 1.22 API	JDK 1.2 compatible Java Virtual Machine (JVM)
JDBC 2.0 Core API	JDK 1.3 compatible JVM
JDBC 2.0 Optional Package	JDK 1.3 compatible JVM Also, the following APIs are required and are supplied with SequeLink Java client: ▪ JDBC 2.0 Optional Package ▪ JNDI 1.2 ▪ JTA 1.0.1
JCA API	JDK 1.3
JDBC 3.0 API	JDK 1.4 or 1.5

To find which version of Java you're running, open a command window (Windows) or Terminal window (Mac OS) and type `java -version`.

Networking requirements

If you'll be accessing a FileMaker data source hosted on another computer, you'll need network access via TCP/IP.

ODBC client driver installation (Windows)

If you have previously installed the ODBC client driver for Windows, uninstall it with Add or Remove Programs before installing an updated version.

To install the ODBC client driver:

You need MDAC 2.8 SP1 (available from www.microsoft.com) to install the ODBC client driver.

1. In the folder \xDBC\ODBC Client Driver Installer, double-click setup.

   The DataDirect SequeLink for ODBC 5.5 - InstallShield Wizard appears.

2. Install the ODBC client driver by following the on-screen instructions.

   The Data Source SyncTool, Data Source SyncTool Administrator, and SequeLink Online Books are not chosen as part of the default installation; you do not need them to use the ODBC client driver.

   By default, the ODBC client driver will be installed in this folder: C:\Program Files\DataDirect\slodbc55. Choose another drive or another folder if you wish.

3. When the installation is complete, click Finish.

   The ODBC client driver, DataDirect 32-BIT SequeLink 5.5, is now available for you to configure for accessing a FileMaker data source.

   Note If you have any trouble installing directly, install the driver instead through Add or Remove Programs in Windows (choose Start menu > Control Panel > Add or Remove Programs).

ODBC client driver installation (Mac OS)

If you have previously installed the ODBC client driver for Mac OS, uninstall it before installing an updated version.

To install the ODBC client driver:

From the folder /xDBC/ODBC Client Driver Installer, copy the file SequeLink.bundle to either your System or User library. If you don't have an /ODBC folder, create one manually:

Library	Copy SequeLink.bundle to this folder:	Use this driver path during configuration:
System	/Library/ODBC	/Library/ODBC/SequeLink.bundle/Contents/MacOS/ivslk20.dylib
User	/Users/<user>/Library/ODBC	/Users/<user>/Library/ODBC/SequeLink.bundle/Contents/MacOS/ivslk20.dylib

This client driver has been tested with ODBC Administrator 1.0.1 (available with Mac OS 10.4).

Important Use the ODBC Administrator bundled with the ODBC driver manager software recommended for your client application.

JDBC client driver installation (Windows and Mac OS)

The installation program and the JDBC client driver work on both Windows and Mac OS. You must have write access to the folder where you're installing the JDBC client driver. By default, the folder that contains the installation program is the installation folder.

If you have previously installed the JDBC client driver, uninstall it before installing an updated version.

To install the JDBC client driver:

1. Open the folder \xDBC\JDBC Client Driver Installer, then double-click sljcinstaller.jar.

   The DataDirect SequeLink for JDBC 5.5 Installer window appears.

   Note Your Java Runtime environment should be associated with .JAR archive files. Other applications may also be associated with .JAR archive files, such as WinZip or Stuffit. If so, they can prevent the JDBC Installer from opening. If the Installer window does not open, start either a command window (Windows) or Terminal window (Mac OS) and change to the JDBC Client Driver folder. From here, you can start the installer by typing the following command: `java -jar sljcinstaller.jar.`

2. Click Next.

   Review the license agreement.

3. If the license agreement is acceptable, select I accept the terms of the license agreement, then click Next.

   A window of installation options appears.

4. Select Install Developer's Tools.

   The tools include JDBCTest, which helps you verify your JDBC connections.

5. Enter an Install Directory.

   - Windows: Enter a path that includes your Java executable file (java.exe).
   - Mac OS: Enter /Library/Java/Extensions (or another location included in the ClassPath of your Java application).

6. Click Next.
7. Confirm your installation selections, then click Install.
8. When the installation is complete, click Finish.

   The JDBC client driver is now available for you to configure for accessing a FileMaker data source.

Configuring client drivers

Before using a client application to access a FileMaker data source, you must configure a client driver for the data source. Configuration settings identify the client driver you're using, the location of the data source, and details on how you intend to connect.

Important When configuring a FileMaker client driver, you must specify `2399` as the port. For ODBC (Windows), you'll specify the port in the ODBC Data Source Administrator. For ODBC (Mac OS), you'll specify the port in the ODBC Administrator. For JDBC, you'll specify the port in the JDBC URL.

For additional information about the ODBC client driver on Windows, choose the Windows Start menu > Programs > DataDirect SequeLink for ODBC 5.5 > Driver Help.

Note The Help system, provided by DataDirect Technologies, describes some functionality beyond the scope of using the ODBC and JDBC client drivers for accessing FileMaker data sources.

Where to go from here

After you install and configure a client driver, you can construct and execute SQL queries to access a FileMaker data source.

Client applications sometimes use different terminology for accessing a data source via ODBC. Many applications have menu items with names such as Get external data or SQL query. Review the documentation or Help that comes with your application for details.

Chapter 3
Using ODBC to share FileMaker data

Use the ODBC client driver to connect to a FileMaker data source from another application. The application that uses the ODBC client driver can directly access the data in a FileMaker database file.

- Windows: The FileMaker ODBC client driver is DataDirect 32-BIT SequeLink 5.5.
- Mac OS: The FileMaker ODBC client driver is ivslk20.dylib.

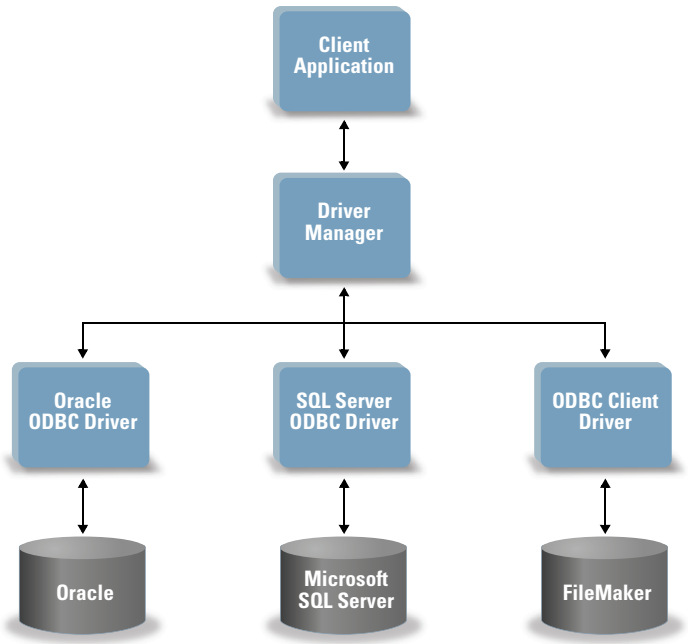
Note You can also use FileMaker Pro as a client application, interacting with records from another data source via ODBC using SQL. See FileMaker Pro Help for details about accessing an external data source via ODBC.

About ODBC

ODBC is an API that enables applications to access data from many database management systems. ODBC gives client applications a common language for interacting with data sources and database services.

All applications that support ODBC recognize a common subset of statements. SQL lets you use other applications (like spreadsheets, word processors, and reporting tools) to view, analyze, and modify FileMaker data. See chapter 5, “Supported standards,” for the SQL statements, functions, and expressions that the ODBC client driver supports.

Your application can talk directly to a FileMaker database file by using the ODBC client driver. Your SQL statements are delivered to the FileMaker host of the database file and the results of those statements are sent back to you. If you use FileMaker Server Advanced to host a FileMaker database file as a data source, the database file can be located on another machine (the server machine) connected to the network, while your client application is located on your machine (the client machine). This is referred to as a client/server configuration.



Using the ODBC client driver

You can use the ODBC client driver with any ODBC-compliant application. Sharing your FileMaker database file as a data source, you can:

- perform mail merges with Microsoft Word
- create charts with Microsoft Excel
- move FileMaker data to a DBMS like Microsoft SQL Server
- further analyze your FileMaker data with query or reporting tools to create charts, construct ad-hoc queries, and perform drill-down analysis
- create a Microsoft Visual Basic application that shares information with FileMaker Pro

To share a FileMaker database file as a data source, use FileMaker Pro to define accounts that need access to the database file. Then, control access to the database file by assigning privilege sets to the accounts, including the extended privilege of access via ODBC/JDBC. Finally, enable the FileMaker Server Advanced or FileMaker Pro host application to share data via ODBC/JDBC. For details, see FileMaker Pro or FileMaker Server Help.

Important The ODBC client driver replaces the FileMaker Pro ODBC driver released with a previous version of FileMaker. If you have previously set up access to a FileMaker data source using the older driver, you'll need to re-define access by using and configuring the new driver.

Overview of accessing a FileMaker database file

From an ODBC-compliant application, you can construct SQL queries to access a FileMaker database file. The ODBC client driver must be installed on the computer generating the SQL query.

To access a FileMaker database file:

1. In FileMaker Pro, review the privilege sets you've assigned to accounts that will access the database file.

   Accounts that need access must use a privilege set with the extended privilege of Access via ODBC/JDBC.
2. Enable the FileMaker Server Advanced (via FileMaker Server Admin Console) or FileMaker Pro host application to share data via ODBC/JDBC.

   FileMaker Server Admin Console: Click ODBC/JDBC then select Enable ODBC/JDBC.

   FileMaker Pro: Choose File menu > Sharing > ODBC/JDBC and set ODBC/JDBC Sharing to On.
3. Make sure the FileMaker database file you want to access is hosted and available.

   If your FileMaker database solution uses more than one FileMaker database file, all of the database files must be on the same computer.
4. Connect to the FileMaker data source.
5. Construct and execute an SQL query in the client application.

   Each FileMaker database file that is open and set up for access is a separate data source (you create a DSN for each FileMaker database file you want to access as a data source).

   Each database can have one or more tables. FileMaker fields are represented as columns. The complete field name, including any non-alphanumeric characters, displays as the column name.

Note In Windows, Microsoft Access can import only 32 or fewer fields at one time via ODBC from a FileMaker database file. If your database file has more than 32 fields, import them in increments of 32.

Accessing a FileMaker database file from a Windows application

Specifying ODBC client driver properties for a FileMaker DSN (Windows)

Create a DSN for each FileMaker database file you want to access as a data source. The DSN identifies the FileMaker ODBC client driver, the location of the FileMaker host application, and the FileMaker database file you're accessing as a data source.

To set up or change ODBC client driver properties:

1. Open the ODBC Data Source Administrator control panel.

   In the Windows Control Panel, choose Administrative Tools > Data Sources (ODBC).

   In Windows XP, Administrative Tools appear in the Performance and Maintenance category. In Windows Vista, Administrative Tools appear in the System and Maintenance category.

2. Click the System DSN tab.

   If you set up your data source as a User DSN or File DSN, click the corresponding tab.

3. Click Add.

   Note If you're changing the properties of an existing data source, select the data source, click Configure, and skip to step 6.

4. Choose DataDirect 32-BIT SequeLink 5.5 from the list of drivers.

   If the driver is not listed in the ODBC Data Source Administrator, look for the (Default) entry of the registry key HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\ODBC Drivers using Regedit (choose Start menu > Run > Regedit). The Data for this entry should be (value not set). If instead you see a blank string, right-click (Default), choose Delete, and click Yes to confirm. This resets the entry to (value not set). Uninstall and reinstall your ODBC client driver to see it in the ODBC Data Source Administrator.

5. Click Finish.

   The DataDirect SequeLink for ODBC Setup dialog appears, with the General tab selected.

6. For Data Source Name, type a name that will be meaningful to others accessing the FileMaker data source.

   An additional Description is optional.

   Be sure Use LDAP is cleared. No translator .DLLs are included with the ODBC client driver (so clicking Translate will not allow you to configure an ODBC translator).

7. For SequeLink Server Host, enter the location of your data source.

   If you're connecting to a FileMaker database file hosted by FileMaker Pro on your local machine, type `localhost` (or `127.0.0.1`).

   If you're connecting to a FileMaker database file hosted by FileMaker Server over a network, type the IP address of FileMaker Server.

8. For SequeLink Server Port, type `2399`.

   Important When configuring a FileMaker data source, you must specify `2399` as the SequeLink Server Port.

9. For Server Data Source, type the filename of the FileMaker database file you're using as a data source (don't type the filename extension).

10. If you've enabled sharing via ODBC/JDBC in the host application, click the button to the right of Server Data Source to display the filenames of currently open FileMaker database files for you to choose from.

    If your database name contains spaces, replace them with the escape characters %20. For example, `serverdatasource=MY%20DATABASE`.

11. Click OK to save your data source information.

    If you're sharing another FileMaker database file, return to step 3 and set up the database file as a data source.

12. Click OK to close the ODBC Data Source Administrator dialog box.

Verifying access via ODBC (Windows)

To verify that you've correctly configured the ODBC client driver to access the FileMaker data source:

1. Open the ODBC Data Source Administrator control panel.

   In the Windows Control Panel, choose Administrative Tools > Data Sources (ODBC).

   In Windows XP, Administrative Tools appear in the Performance and Maintenance category.

2. Click the System DSN tab.

   If you set up your data source as a User DSN or File DSN, click the corresponding tab.

3. Choose the FileMaker data source that you previously configured.

   The data source name you originally entered appears under Name, and DataDirect 32-BIT SequeLink 5.5 appears as the Driver.

4. Click Configure.

   The DataDirect SequeLink for ODBC Setup dialog box appears.

5. Click Test Connect.

   You are prompted to enter your FileMaker account name (in Database User Name) and password (in Database Password).

If the connection is OK, you'll receive the message Connection test was successful. If the connection fails:

- Make sure the FileMaker database file is hosted and available.
- Update or correct your connection information.
- Make sure your FileMaker account uses a privilege set with the extended privilege of Access via ODBC/JDBC.
- Verify that the FileMaker Pro or FileMaker Server host application has been set up for sharing via ODBC/JDBC.

Accessing a FileMaker database file from a Mac OS application

If you build custom applications, use version 3.52.1 of the ODBC headers and libraries. Applications built with version 3.51 might not be able to load the client driver.

Configuring the ODBC client driver (Mac OS)

Configure the client driver using the ODBC Administrator bundled with the ODBC driver manager software recommended for your client application.

The client driver has been tested with ODBC Administrator 1.0.1 (available with Mac OS 10.4).

When configuring the client driver, you'll be prompted to provide a brief description and the path to the driver file:

If you copied SequeLink.bundle to this library:	Use this driver path during configuration:
/Library/ODBC	/Library/ODBC/SequeLink.bundle/Contents/MacOS/ivslk20.dylib
/Users/<user>/Library/ODBC	/Users/<user>/Library/ODBC/SequeLink.bundle/Contents/MacOS/ivslk20.dylib

The ODBC Administrator also allows you to optionally define keywords and a Setup File path, but the client driver does not need that information.

Specifying ODBC client driver properties for a FileMaker DSN (Mac OS)

Create a DSN for each FileMaker database file you want to access as a data source. The DSN identifies the FileMaker ODBC client driver, the location of the FileMaker host application, and the FileMaker database file you're accessing as a data source.

Important The ODBC client driver for Mac OS does not support upper-ASCII, double-byte, or Japanese characters in database names or table names. If your FileMaker database file uses these characters, create a second database and use only ASCII characters for the filename and table names. In the second database, create a data source reference that points to the data in your original database file. Share both files with ODBC/JDBC, but use the second database file when defining the DSN.

Additionally, you'll need to specify these keyword values for the DSN:

Keyword	Value
Host	If you're connecting to a FileMaker database file hosted by FileMaker Pro on your local machine, type `localhost` (or `127.0.0.1`). If you're connecting to a FileMaker database file hosted by FileMaker Server over a network, type the IP address or hostname of FileMaker Server.
Port	Type `2399`.
ServerDataSource	Type the filename of the FileMaker database file you're using as a data source (don't type the filename extension). If your database name contains spaces, replace them with the escape characters `%20`. For example, `ServerDataDource=MY%20DATABASE`. Double-byte characters are not supported.

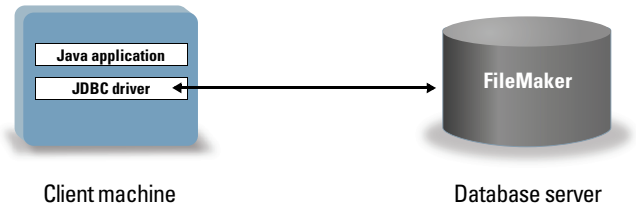
Chapter 4
Using JDBC to share FileMaker data

If you're a Java programmer, you can use the JDBC client driver with any Rapid Application Development (RAD) tool to visually create a Java application or applet that connects to a FileMaker data source. The Java application or applet that uses the JDBC client driver can directly access the data in a FileMaker database file.

About JDBC

JDBC is a Java API for executing SQL statements, the standard language for accessing relational databases. JDBC is a name and not an acronym — although it is thought of as standing for "Java Database Connectivity" because it is the Java equivalent for ODBC. JDBC is a low-level interface, which means that it is used to call SQL commands directly. It is also designed to be used as a base for higher level interfaces and tools.

Your Java applet or application can talk directly to a FileMaker database file by using the JDBC client driver. Your SQL statements are delivered to the FileMaker host of the database file and the results of those statements are sent back to you. If you use FileMaker Server to host, the FileMaker database file you're using as a data source can be located on another machine (the server machine) connected to the network, while your Java applet or client application is located on your machine (the client machine). This is referred to as a client/server configuration.



Using the JDBC client driver

You can use the JDBC client driver with a Java compiler or RAD tool to connect with your database while you build the code for your Java application or applet. After the Java application or applet has been created, the JDBC client driver must be present with the files or included within the code in order for the application or applet to communicate with the database.

To use the JDBC client driver, your Java application or applet must register the driver with the JDBC driver manager and you must specify the correct JDBC URL from within the application or applet. You need the JDBC URL to make the connection to the database.

About the JDBC client driver

The JDBC client driver provides partial support for the JDBC 3.0 specification. The following features are not supported by FileMaker:

- Savepoint support
- Retrieval of auto-generated keys
- Passing parameters to a callable statement object by name
- Holdable cursor support
- Making internal updates to the data in Blob and Clob objects
- Retrieving and updating the object referenced by a Ref object
- Updating of columns containing BLOB, CLOB, ARRAY and REF data types
- Boolean data type
- DATALINK data type
- Transform groups and type mapping
- Relationship between the JDBC SPI and the Connector architecture

For additional details, see www.filemaker.com/support/technologies. Also see the *SequeLink Developer's Reference* available at www.datadirect.com for information about JDBC compatibility and developing JDBC applications for SequeLink.

The JDBC client driver has been tested against the Java Development Kit (JDK) 1.4 and 1.5. It is a Type 4 driver — a native protocol, pure Java driver that converts JDBC calls directly into the network protocol used by FileMaker. This type of driver offers all the advantages of Java including automatic installation (for example, downloading the JDBC driver with an applet that uses it). The driver works with JDK 1.3 and Java 2 as long as you only use JDBC 1.2 calls in a Java 2 environment.

The driver class and main entry point for the driver is named:

com.ddtek.jdbc.sequelink.SequeLinkDriver

Important The JDBC client driver replaces the FileMaker JDBC driver released with a previous version of FileMaker. If you have previously set up access to a FileMaker data source using the older driver, you'll need to re-define access by using and configuring the new driver.

Using a JDBC URL to connect to your database

In Java, most resources are accessed through URLs (Uniform Resource Locators). A JDBC URL is used to identify the database so the JDBC client driver can recognize and establish a connection with the database.

The JDBC URL consists of three main parts separated by colons:

jdbc:<subprotocol>:<subname>

The first part in the JDBC URL is always the JDBC protocol ("jdbc"). The *subprotocol* is the driver name or the mechanism that supports multiple drivers. For the JDBC client driver, the subprotocol is `sequelink`. The *subname* is the IP address of the machine that is hosting the FileMaker data source.

Registering the JDBC client driver and connecting to a FileMaker data source (an example)

Here is a snippet of a JDBC client application that:

1. Registers the JDBC client driver with the JDBC driver manager.
2. Establishes a connection with the FileMaker data source (the JDBC URL is jdbc:sequelink://17.184.17.170:2399).
3. Returns error codes.

```
import java.sql.*;
class FMPJDBCTest
{
    public static void main(String[ ] args)
    {
        // register the JDBC client driver
        try {
            Driver d =
        (Driver)Class.forName("com.ddtek.jdbc.sequelink.SequeLinkDriver").newInstance();
        } catch(Exception e) {
            System.out.println(e);
        }
        // establish a connection to FileMaker
        Connection con;
        try {
            con =
        DriverManager.getConnction("jdbc:sequelink://17.184.17.170:2399;
        user=some user;password=some password;serverDataSource=database");
        } catch(Exception e) + ";serverdatasource=" + dbName{
            System.out.println(e);
        }
        // get connection warnings + ";serverDataSource=" + dbName
        SQLWarning warning = null;
        try {
            warning = con.getWarnings();
            if (warning == null) {
                    System.out.println("No warnings");
                    return;
            }
            while (warning != null) {
                    System.out.println("Warning: "+warning);
                    warning = warning.getNextWarning();
            }
        } catch (Exception e) {
            Sysem.out.println(e);
        }
    }
}þ
```

Note This example is not meant to be compiled.

Specifying driver properties in the URL subname

Specify the user and password driver properties in the subname of the JDBC URL. These are the properties that could be passed to the connection when calling the DriverManager.getConnection method via the Properties parameter.

- user: an account in the FileMaker database file that uses a privilege set with the extended privilege Access via ODBC/JDBC
- password: the password for the account in the FileMaker database file

Basic JDBC URL connection

Format: `jdbc:sequelink://<sequelink host IP address>:<port>`

This is the URL to connect to the FileMaker database file with no passwords. The port number is always 2399 (you can't change the JDBC sharing to a different port).

If you are executing the JDBC URL connection on the same machine where the file is located, you can use jdbc:sequelink://localhost:2399

Example: `jdbc:sequelink://17.184.17.170:2399`

JDBC URL connection with user name and password defined in the URL

Format:

```
jdbc:sequelink://<sequelink host IP
address>:<port>;user=<userName>;password=<password>
```

Example: `jdbc:sequelink://17.184.17.170:2399;user=phil;password=jsp`

JDBC URL connection with the database name specified in the URL

Format:

```
jdbc:sequelink://<sequelink host IP address>:<port>;
serverDataSource=<databasename>
```

Example: `jdbc:sequelink://17.184.17.170:2399;serverDataSource=publications`

If your database name contains spaces, replace them with the escape characters %20.

Example: `jdbc:sequelink://17.184.17.170:2399;serverDataSource=MY%20DATABASE`

JDBC URL connection with the database name, user name, and password specified in the URL

Format 1 (using the data store user name and password): `jdbc:sequelink://<sequelink host IP address>:<port>;serverDataSource=<databasename>;DBUser=<databaseusername>;DBPassword=<databasepassword>`

Format 2 (using the host user name and password): `jdbc:sequelink://<sequelink host IP address>:<port>;serverDataSource=<databasename>;HUser=<username>;HPassword=<password>`

If your FileMaker database solution uses many FileMaker database files, create an additional database file that contains all the necessary external data source references, table occurrences, and relationships for your solution. Then define this additional database file as your data source in the JDBC URL. All of the FileMaker database files must be on the same computer.

Note Secure Socket Layer (SSL) encryption is not supported. To create a more secure JDBC solution, set up an environment such as .ASP or .JSP, where the web server is hosting via https and communicating with the FileMaker database file behind a firewall.

Verifying access via JDBC

When you install the JDBC client driver, you have the option of installing JDBCTest to help verify your connections. When installing the JDBC client driver, select Install Developer's Tools to get JDBCTest. See "JDBC client driver installation (Windows and Mac OS)" on page 13 for more information.

When verifying access to a FileMaker database file via JDBC, make sure:

- The FileMaker database file is hosted and available.
- Your FileMaker account uses a privilege set with the extended privilege of Access via ODBC/JDBC.
- The FileMaker Pro or FileMaker Server Advanced host application has been set up for sharing via ODBC/JDBC.

  To share a FileMaker database file as a data source, use FileMaker Pro to define accounts that need access to the database file. Then, control access to the database file by assigning privilege sets to the accounts, including the extended privilege of access via ODBC/JDBC. Finally, enable the FileMaker Server Advanced or FileMaker Pro host application to share data via ODBC/JDBC. For details, see FileMaker Pro Help.
- The JDBC client driver registration and the JDBC URL are correct (the driver can be included inside the Java Application or located on the client machine).

For additional information on using JDBC to share FileMaker data, see www.filemaker.com/support/technologies.

Chapter 5
Supported standards

This chapter describes the SQL statements and constructs supported by the FileMaker ODBC and JDBC client drivers. Use the client drivers to access a FileMaker database solution from an ODBC- or JDBC-compliant application. The FileMaker database solution can be hosted by either FileMaker Pro or FileMaker Server Advanced.

The ODBC client driver supports ODBC 3.5 Level 1 with some features of Level 2. The JDBC client driver provides partial support for the JDBC 3.0 specification. See www.filemaker.com/support/technologies and the *SequeLink Developer's Reference* available at www.datadirect.com for more information. The ODBC and JDBC client drivers support SQL-92 entry-level conformance, with some SQL-92 intermediate features.

Support for Unicode characters

The ODBC and JDBC client drivers support the Unicode API. However, if you're creating a custom application that uses the client drivers, use ASCII for field names, table names, and filenames (in case a non-Unicode query tool or application is used).

Note To insert and retrieve Unicode data, use SQL_C_WCHAR (the SQL_C_BINARY data type is not supported).

SQL statements

The ODBC and JDBC client drivers provide support for the following SQL statements:

SELECT (see below)	DELETE (page 33)	INSERT (page 33)	UPDATE (page 34)
CREATE TABLE (page 35)	ALTER TABLE (page 35)	CREATE INDEX (page 36)	DROP INDEX (page 36)

The client drivers also support FileMaker data type mapping to ODBC SQL and JDBC SQL data types. See appendix A, "Mapping FileMaker fields to ODBC data types" and appendix B, "Mapping FileMaker fields to JDBC data types" for data type conversions. For more information on constructing SQL queries, refer to a third-party book.

Note The ODBC and JDBC client drivers recognize only the first repetition in a repeating field. Also, the drivers do not support portals in FileMaker Pro.

SELECT statement

Use the SELECT statement to specify which columns you're requesting. Follow the SELECT statement with the column expressions (similar to field names) you want to retrieve (for example, `last_name`). Expressions can include mathematical operations or string manipulation (for example, `SALARY * 1.05`).

The SELECT statement can use a variety of clauses:

```
SELECT [DISTINCT] {* | column_expression [[AS] column_alias],...}
FROM table_name [table_alias], ...
[ WHERE expr1 rel_operator expr2 ]
[ GROUP BY {column_expression, ...} ]
[ HAVING expr1 rel_operator expr2 ]
[ UNION [ALL] (SELECT...) ]
[ ORDER BY {sort_expression [DESC | ASC]}, ... ]
[ FOR UPDATE [OF {column_expression, ...}] ]
```

Items in brackets are optional.

Note SELECT * on larger databases and SELECT statements that use table aliases or literals in the projection list might not function correctly. To avoid potential confusion, avoid wildcards and specify table and column names without aliases.

`column_alias` can be used to give the column a more descriptive name, or to abbreviate a longer column name. For example, to assign the alias `department` to the column `dept`:

```
SELECT dept AS department FROM emp
```

Field names can be prefixed with the table name or the table alias. For example, EMP.LAST_NAME or E.LAST_NAME, where E is the alias for the table EMP.

The DISTINCT operator can precede the first column expression. This operator eliminates duplicate rows from the result of a query. For example:

```
SELECT DISTINCT dept FROM emp
```

SQL clauses

The ODBC and JDBC client drivers provide support for the following SQL clauses.

Use this SQL clause	To
FROM (see below)	Indicate which tables are used in the SELECT statement.
WHERE (page 29)	Specify the conditions that records must meet to be retrieved (like a FileMaker Pro find request).
GROUP BY (page 29)	Specify the names of one or more fields by which the returned values should be grouped. This clause is used to return a set of aggregate values by returning one row for each group (like a FileMaker Pro subsummary).
HAVING (page 30)	Specify conditions for groups of records (for example, display only the departments that have salaries totaling more than $200,000). This clause is only valid if you have already defined a GROUP BY clause.
UNION (page 29)	Combine the results of two or more SELECT statements into a single result.
ORDER BY (page 30)	Indicate how the records are sorted
FOR UPDATE (page 31)	To perform Positioned Updates or Positioned Deletes via SQL cursors

Note If you attempt to retrieve data from a table with no columns, the SELECT statement fails.

FROM clause

The FROM clause indicates the tables that are used in the SELECT statement. The format is:

```
FROM table_names [table_alias]
```

`table_names` can be one or more simple table names in the current working directory or complete pathnames.

`table_alias` can be used to give the table a more descriptive name, or to abbreviate a longer table name.

Field names can be prefixed with the table name or the table alias. For example, given the table specification `FROM employee E`, you can refer to the LAST_NAME field as E.LAST_NAME. Table aliases must be used if the SELECT statement joins a table to itself. For example:

```
SELECT * FROM employee E, employee F WHERE E.manager_id = F.employee_id
```

The equal sign (=) includes only matching rows in the results.

If you are joining more than one table, and you want to discard all rows that don't have corresponding rows in both source tables, you can use INNER JOIN. For example:

```
SELECT *
FROM Salespeople INNER JOIN Sales_Data
ON Salespeople.Salesperson_ID = Sales_Data.Salesperson_ID
```

Note OUTER JOIN is not currently supported.

WHERE clause

The WHERE clause specifies the conditions that records must meet to be retrieved. The WHERE clause contains conditions in the form:

```
WHERE expr1 rel_operator expr2
```

`expr1` and `expr2` can be field names, constant values, or expressions.

`rel_operator` is the relational operator that links the two expressions. For example, the following SELECT statement retrieves the names of employees who make $20,000 or more.

```
SELECT last_name,first_name FROM emp WHERE salary >= 20000
```

Note If you use fully qualified names in the SELECT (projection) list, you must also use fully qualified names in the related WHERE clause.

GROUP BY clause

The GROUP BY clause specifies the names of one or more fields by which the returned values should be grouped. This clause is used to return a set of aggregate values. It has the following format:

```
GROUP BY column_expressions
```

`column_expressions` must match the column expression used in the SELECT clause. A column expression can be one or more field names of the database table separated by commas, or one or more expressions separated by commas.

The following example sums the salaries in each department.

```
SELECT dept_id, SUM (salary) FROM emp GROUP BY dept_id
```

This statement returns one row for each distinct department ID. Each row contains the department ID and the sum of the salaries of the employees in the department.

HAVING clause

The HAVING clause enables you to specify conditions for groups of records (for example, display only the departments that have salaries totaling more than $200,000). This clause is valid only if you have already defined a GROUP BY clause. It has the following format:

```
HAVING expr1 rel_operator expr2
```

`expr1` and `expr2` can be field names, constant values, or expressions. These expressions do not have to match a column expression in the SELECT clause.

`rel_operator` is the relational operator that links the two expressions. The following example returns only the departments whose sums of salaries are greater than $200,000:

```
SELECT dept_id, SUM (salary) FROM emp
GROUP BY dept_id HAVING SUM (salary) > 200000
```

UNION operator

The UNION operator combines the results of two or more SELECT statements into a single result. The single result is all of the returned records from the SELECT statements. By default, duplicate records are not returned. To return duplicate records, use the ALL keyword (UNION ALL). The format is:

```
SELECT statement UNION [ALL] SELECT statement
```

When using the UNION operator, the select lists for each SELECT statement must have the same number of column expressions, with the same data types, and must be specified in the same order. For example:

```
SELECT last_name, salary, hire_date FROM emp UNION SELECT name, pay,
birth_date FROM person
```

This example has the same number of column expressions, and each column expression, in order, has the same data type.

The following example is not valid because the data types of the column expressions are different (SALARY from EMP has a different data type than LAST_NAME from RAISES). This example has the same number of column expressions in each SELECT statement, but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp UNION SELECT salary, last_name FROM raises
```

ORDER BY clause

The ORDER BY clause indicates how the records are to be sorted. The format is:

```
ORDER BY {sort_expression [DESC | ASC]}, ...
```

`sort_expression` can be field names, expressions, or the positional number of the column expression to use. The default is to perform an ascending (ASC) sort.

For example, to sort by `last_name` then by `first_name`, you could use either of the following SELECT statements:

```
SELECT emp_id, last_name, first_name FROM emp ORDER BY last_name, first_name
```

or

```
SELECT emp_id, last_name, first_name FROM emp ORDER BY 2,3
```

In the second example, `last_name` is the second column expression following SELECT, so ORDER BY 2 sorts by `last_name`.

FOR UPDATE clause

The FOR UPDATE clause performs Positioned Updates or Positioned Deletes via SQL cursors. The format is:

```
FOR UPDATE [OF column_expressions]
```

`column_expressions` is a list of field names in the database table that you intend to update, separated by a comma. `column_expressions` is optional.

The following example returns all records in the employee database that have a SALARY field value of more than $20,000. When each record is fetched, it is locked. If the record is updated or deleted, the lock is held until you commit the change. Otherwise, the lock is released when you fetch the next record.

```
SELECT * FROM emp WHERE salary > 20000 FOR UPDATE OF last_name, first_name,
salary
```

Additional examples:

Using	Sample SQL
text constant	`SELECT 'CatDog' FROM Salespeople`
numeric constant	`SELECT 999 FROM Salespeople`
date constant	`SELECT DATE '2004-06-05' FROM Salespeople`
time constant	`SELECT TIME '02:49:03' FROM Salespeople`
timestamp constant	`SELECT TIMESTAMP '2004-06-05 02:49:03' FROM Salespeople`
text column	`SELECT Company_Name FROM Sales_Data` `SELECT DISTINCT Company_Name FROM Sales_Data`
numeric column	`SELECT Amount FROM Sales_Data` `SELECT DISTINCT Amount FROM Sales_Data`
date column	`SELECT Date_Sold FROM Sales_Data` `SELECT DISTINCT Date_Sold FROM Sales_Data`
time column	`SELECT Time_Sold FROM Sales_Data` `SELECT DISTINCT Time_Sold FROM Sales_Data`
timestamp column	`SELECT Timestamp_Sold FROM Sales_Data` `SELECT DISTINCT Timestamp_Sold FROM Sales_Data`
BLOB[a] column	`SELECT Company_Brochures FROM Sales_Data` `SELECT GETAS(Company_Logo, 'JPEG') FROM Sales_Data`
Wildcard *	`SELECT * FROM Salespeople` `SELECT DISTINCT * FROM Salespeople`

a. A BLOB is a FileMaker database file container field.

Notes from the examples

A `column` is a reference to a field in the FileMaker database file (the field can contain many distinct values).

The asterisk (*) wildcard character is shorthand for "everything". For the example `SELECT * FROM Salespeople`, the result is all the rows in the `Salespeople` table. For the example `SELECT DISTINCT * FROM Salespeople`, the result is all the unique rows in the `Salespeople` table (no duplicates).

Note SELECT * statements on larger databases might not function correctly. To avoid potential confusion, avoid wildcards and specify table and column names (without aliases).

Retrieving the contents of a container field: CAST() function and GetAs() function

You can retrieve binary data, file reference information, or data of a specific file type from a container field.

To retrieve binary data, use a standard SELECT statement. For example:

```
SELECT Company_Brochures FROM Sales_Data
```

If file or JPEG data exists, the SELECT statement retrieves the data in binary form; otherwise, the SELECT statement returns <null>.

To retrieve file reference information (such as the file path), use the CAST function with a SELECT statement. For example:

```
SELECT CAST(Company_Brochures AS VARCHAR(NNN)) FROM Sales_Data
```

In this example, if you:

- Inserted a file into the container field using FileMaker Pro but stored only a reference to the file, the SELECT statement retrieves the file reference information as type SQL_VARCHAR.
- Inserted the contents of a file into the container field using FileMaker Pro, the SELECT statement retrieves the name of the file.
- Imported a file into the container field from another application, the SELECT statement displays '?' (the file displays as Untitled.dat in FileMaker Pro).

To retrieve data of a specific file type from a container field, use the GetAs function and specify the file's type. For example:

```
SELECT GetAs(Company_Logo, 'JPEG') FROM Sales_Data
```

The possible file types (case sensitive) you can retrieve from a container field in a FileMaker database file are:

File type	Description	File type	Description
'EMBO'	OLE container data	'PDF '	Portable Document Format
'EMF+'	Windows Enhanced Metafile Plus	'PICT'	Mac OS (does not have 512-byte file-based header)
'EPS '	Embedded PostScript	'PNGf'	Bitmap image format
'FILE'	Result of an Insert File command	'PNTG'	MacPaint
'FPix'	Flash (FPX)	'qtif'	QuickTime image file
'FORK'	Resource fork (Mac OS)	'.SGI'	Generic bitmap format
'GIFf'	Graphics Interchange Format	'snd '	Standard sound (Mac OS raw format)
'JPEG'	Photographic images	'TIFF'	Raster file format for digital images
'JP2 '	JPEG 2000	'TPIC'	Targa
'META'	Windows Metafile (enhanced)	'XMLO'	Layout objects
'METO'	Windows Metafile (original)	'8BPS'	PhotoShop (PSD)
'moov'	Old QuickTime format (Mac OS)		

DELETE statement

Use the DELETE statement to delete records from a database table. The format of the DELETE statement is:

```
DELETE FROM table_name [ WHERE { conditions } ]
```

Note The WHERE clause determines which records are to be deleted. If you don't include the WHERE keyword, all records in the table are deleted (but the table is left intact).

An example of a DELETE statement on the Employee table is:

```
DELETE FROM emp WHERE emp_id = 'E10001'
```

Each DELETE statement removes every record that meets the conditions in the WHERE clause. In this case, every record having the employee ID E10001 is deleted. Because employee IDs are unique in the Employee table, only one record is deleted.

INSERT statement

Use the INSERT statement to create records in a database table. You can specify either:

- A list of values to be inserted as a new record
- A SELECT statement that copies data from another table to be inserted as a set of new records

The format of the INSERT statement is:

```
INSERT INTO table_name [(column_name, ...)] VALUES (expr, ...)
```

`column_name` is an optional list of column names that provides the name and order of the columns whose values are specified in the VALUES clause. If you omit `column_name`, the value expressions (`expr`) must provide values for all columns defined in the table and must be in the same order that the columns are defined for the table.

`expr` is the list of expressions giving the values for the columns of the new record. Usually the expressions are constant values for the columns (but they can also be a subquery). You must enclose character string values in pairs of single quotation marks ('). To include a single quotation mark in a character string value enclosed by single quotation marks, use two single quotation marks together (for example, 'Don''t'). Date, time, and timestamp values must be enclosed in braces {}. Logical values that are characters must be enclosed in periods (for example, .T. or .F.). Subqueries must be enclosed in parentheses.

The following example inserts a list of expressions:

```
INSERT INTO emp (last_name, first_name, emp_id, salary, hire_date)
VALUES ('Smith', 'John', 'E22345', 27500, {6/5/2004})
```

Each INSERT statement adds one record to the database table. In this case a record has been added to the employee database table, EMP. Values are specified for five columns. The remaining columns in the table are assigned a blank value, meaning Null.

Note In container fields, you can INSERT only text.

The SELECT statement is a query that returns values for each `column_name` value specified in the column name list. Using a SELECT statement instead of a list of value expressions lets you select a set of rows from one table and insert it into another table using a single INSERT statement.

Here's an example of an INSERT statement that uses a SELECT statement:

```
INSERT INTO emp1 (first_name, last_name, emp_id, dept, salary)
SELECT first_name, last_name, emp_id, dept, salary from emp
WHERE dept = 'D050'
```

In this type of INSERT statement, the number of columns to be inserted must match the number of columns in the SELECT statement. The list of columns to be inserted must correspond to the columns in the SELECT statement just as it would to a list of value expressions in the other type of INSERT statement. For example, the first column inserted corresponds to the first column selected; the second inserted to the second, and so on.

The size and data type of these corresponding columns must be compatible. Each column in the SELECT list should have a data type that the ODBC or JDBC client driver accepts on a regular INSERT/UPDATE of the corresponding column in the INSERT list. Values are truncated when the size of the value in the SELECT list column is greater than the size of the corresponding INSERT list column.

The SELECT statement is evaluated before any values are inserted.

UPDATE statement

Use the UPDATE statement to change records in a database table. The format of the UPDATE statement is:

```
UPDATE table_name SET column_name = expr, ... [ WHERE { conditions } ]
```

`column_name` is the name of a column whose value is to be changed. Several columns can be changed in one statement.

`expr` is the new value for the column. Usually the expressions are constant values for the columns (but they can also be a subquery). You must enclose character string values in pairs of single quotation marks ('). To include a single quotation mark in a character string value enclosed by single quotation marks, use two single quotation marks together (for example, 'Don''t'). Date, time, and timestamp values must be enclosed in braces {}. Logical values that are characters must be enclosed in periods (for example, .T. or .F.). Subqueries must be enclosed in parentheses.

The WHERE clause is any valid clause. It determines which records are updated.

An example of an UPDATE statement on the Employee table is:

```
UPDATE emp SET salary=32000, exempt=1 WHERE emp_id = 'E10001'
```

The UPDATE statement changes every record that meets the conditions in the WHERE clause. In this case the salary and exempt status are changed for all employees having the employee ID E10001. Because employee IDs are unique in the Employee table, only one record is updated.

Here's an example using a subquery:

```
UPDATE emp SET salary = (SELECT avg(salary) from emp) WHERE emp_id = 'E10001'
```

In this case, the salary is changed to the average salary in the company for the employee having employee ID E10001.

Note In container fields, you can UPDATE only with text.

CREATE TABLE statement

Use the CREATE TABLE statement to create a table in a database file. The format of the CREATE TABLE statement is:

```
CREATE TABLE table_name table_element_list [NOT NULL]
```

Within the statement, you specify the name and data type of each column.

`table_name` and `table_element_list` have a 100 character limit. Defining a column to be `NOT NULL` automatically selects the Not Empty Validation Option for the corresponding field in the FileMaker database file. The field is flagged as a Required Value in the Fields tab of the Manage Database dialog box in FileMaker Pro.

Examples

Using	Sample SQL
text column	`CREATE TABLE T1 (C1 VARCHAR, C2 VARCHAR (50), C3 VARCHAR (1001), C4 VARCHAR (500276))`
text column, NOT NULL	`CREATE TABLE T1NN (C1 VARCHAR NOT NULL, C2 VARCHAR (50) NOT NULL, C3 VARCHAR (1001) NOT NULL, C4 VARCHAR (500276) NOT NULL)`
numeric column	`CREATE TABLE T2 (C1 DECIMAL, C2 DECIMAL (10,0), C3 DECIMAL (7539,2), C4 DECIMAL (497925,301))`
date column	`CREATE TABLE T3 (C1 DATE, C2 DATE, C3 DATE, C4 DATE)`
time column	`CREATE TABLE T4 (C1 TIME, C2 TIME, C3 TIME, C4 TIME)`
timestamp column	`CREATE TABLE T5 (C1 TIMESTAMP, C2 TIMESTAMP, C3 TIMESTAMP, C4 TIMESTAMP)`
BLOB column	`CREATE TABLE T6 (C1 BLOB, C2 BLOB, C3 BLOB, C4 BLOB)`

ALTER TABLE statement

Use the ALTER TABLE statement to change the structure of an existing table in a database file. You can modify only one column in each statement. The formats of the ALTER TABLE statement are:

```
ALTER TABLE table_name ADD [COLUMN] column_definition
ALTER TABLE table_name DROP [COLUMN] unqualified_column_name
```

You must know the table's structure and how you want to modify it before using the ALTER TABLE statement.

Examples

To	Sample SQL
add columns	`ALTER TABLE Salespeople ADD C1 VARCHAR`
remove columns	`ALTER TABLE Salespeople DROP C1`

CREATE INDEX statement

Use the CREATE INDEX statement to speed searches in your database file. The format of the CREATE INDEX statement is:

```
CREATE INDEX [ index_name ][ON] table_name.column_name
```

CREATE INDEX is supported for a single column (multi-column indexes are not supported). Indexes are not allowed on columns that correspond to container field types, summary fields, fields that have the global storage option, or unstored calculation fields in a FileMaker database file.

Creating an index for a text column automatically selects the Storage Option of Minimal in Indexing for the corresponding field in the FileMaker database file. Creating an index for a non-text column (or a column formatted as Japanese text) automatically selects the Storage Option of All in Indexing for the corresponding field in the FileMaker database file.

Creating an index for any column automatically selects the Storage Option of Automatically create indexes as needed in Indexing for the corresponding field in the FileMaker database file.

Example

```
CREATE INDEX myIndex ON Salespeople.Salesperson_ID
```

DROP INDEX statement

Use the DROP INDEX statement to remove an index from a database file. The format of the DROP INDEX statement is:

```
DROP INDEX [ON] table_name.column_name
```

Remove an index when your database file is too large, or you don't often use a field in queries.

If your queries are experiencing poor performance, and you're working with an extremely large FileMaker database file with many indexed text fields, consider dropping the indexes from some fields. Also consider dropping the indexes from fields that you rarely use in SELECT statements.

Dropping an index for any column automatically selects the Storage Option of None and clears Automatically create indexes as needed in Indexing for the corresponding field in the FileMaker database file.

The PREVENT INDEX CREATION attribute is not supported.

Example

```
DROP INDEX ON Salespeople.Salesperson_ID
```

SQL aggregate functions

Aggregate functions return a single value from a set of records. You can use an aggregate function as part of a SELECT statement, with a field name (for example, AVG(SALARY)), or in combination with a column expression (for example, AVG(SALARY * 1.07)).

You can precede the column expression with the DISTINCT operator to eliminate duplicate values. For example:

```
COUNT (DISTINCT last_name)
```

In this example, only unique last name values are counted.

Important Use uppercase letters for SQL function names (some are case sensitive).

Aggregate function	Returns
SUM	The total of the values in a numeric field expression. For example, SUM(SALARY) returns the sum of all salary field values.
AVG	The average of the values in a numeric field expression. For example, AVG(SALARY) returns the average of all salary field values.
COUNT	The number of values in any field expression. For example, COUNT(NAME) returns the number of name values. When using COUNT with a field name, COUNT returns the number of non-null field values. A special example is COUNT(*), which returns the number of records in the set, including records with null values.
MAX	The maximum value in any field expression. For example, MAX(SALARY) returns the maximum salary field value.
MIN	The minimum value in any field expression. For example, MIN(SALARY) returns the minimum salary field value.

Examples

```
SELECT SUM (Sales_Data.Amount) AS agg FROM Sales_Data

SELECT AVG (Sales_Data.Amount) AS agg FROM Sales_Data

SELECT COUNT (Sales_Data.Amount) AS agg FROM Sales_Data

SELECT MAX (Sales_Data.Amount) AS agg FROM Sales_Data WHERE
Sales_Data.Amount < 3000

SELECT MIN (Sales_Data.Amount) AS agg FROM Sales_Data WHERE
Sales_Data.Amount < 3000
```

SQL expressions

Use expressions in WHERE, HAVING, and ORDER BY clauses of SELECT statements to form detailed and sophisticated database queries.Valid expression elements are:

Field names	Numeric operators	Relational operators
Constants and literals	Character operators	Logical operators
Exponential notation	Date operators	Functions

Field names

The most common expression is a simple field name, such as `calc` or `Sales_Data.Invoice_ID`.

Constants and literals

Constants are values that do not change. For example, in the expression PRICE * 1.05, the value 1.05 is a constant. Or you might assign a value of 30 to the constant Number_Of_Days_In_June.

A literal is another kind of constant; but instead of having an assigned value, the literal itself is the value, such as 'Paris' or '14:35:10'. A literal is a “what you see is what you get” constant.

You must enclose character constants (such as literals) in pairs of single quotation marks ('). To include a single quotation mark in a character constant enclosed by single quotation marks, use two single quotation marks together (for example, 'Don''t').

You must enclose date, time, and timestamp constants in braces ({}), for example, {D '2005-06-05'}, {14:35:10}, and {TS '2005-06-05 14:35:10'}. The one exception: SQL-92 syntax requires ISO date and time formats with no brackets:

- DATE 'YYYY-MM-DD'
- TIME 'HH:MM:SS'
- TIMESTAMP 'YYYY-MM-DD HH:MM:SS'

Constant	Acceptable syntax (examples)
Text	'Paris'
Number	1.05
Date	DATE '2005-06-05' { D '2005-06-05' } { 06/05/2005 } { 06/05/05 }
Time	TIME '14:35:10' { T '14:35:10' } {14:35:10}
Timestamp	TIMESTAMP '2005-06-05 14:35:10' { TS '2005-06-05 14:35:10'} { 06/05/2005 14:35:10 } {2005-06-05 14:35:10} { 06/05/05 14:35:10 } Make sure Strict data type: 4-Digit Year Date is not selected as a validation option in the FileMaker database file for a field using this 2-digit year syntax.

When entering date and time values, match the format of the database file locale. For example, if the database was created on an Italian language system, use Italian date and time formats.

Logical values that are characters must be enclosed in periods. The logical constants are .T. and 1 for True and .F. and 0 for False. For portability, use 1 and 0.

Exponential/scientific notation

You can include exponential notation.

Example

```
SELECT column1, 3.4E+7 FROM table1 WHERE calc < 3.4E-6 * column2
```

Numeric operators

You can include the following operators in number expressions: +, -, *, /, and ^ or ** (exponentiation). You can precede numeric expressions with a unary plus (+) or minus (-).

Character operators

You can concatenate characters.

Examples

In the following examples, last_name is 'JONES ' and first_name is 'ROBERT ':

Operator	Concatenation	Example	Result
+	Keep trailing blank characters	first_name + last_name	'ROBERT JONES '
-	Move trailing blank characters to the end	first_name - last_name	'ROBERTJONES '

Date operators

You can modify dates.

Examples

In the following examples, hire_date is {01/30/2004}.

Operator	Effect on date	Example	Result
+	Add a number of days to a date	hire_date + 5	{02/04/2004}
-	Find the number of days between two dates, or subtract a number of days from a date	hire_date - {01/01/2004} hire_date - 10	29 {01/20/2004}

Additional examples:

```
SELECT Date_Sold, Date_Sold + 30 AS agg FROM Sales_Data
SELECT Date_Sold, Date_Sold - 30 AS agg FROM Sales_Data
```

Relational operators

Operator	Meaning
=	Equal
<>	Not equal
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
LIKE	Matching a pattern
NOT LIKE	Not matching a pattern
IS NULL	Equal to Null
IS NOT NULL	Not equal to Null
BETWEEN	Range of values between a lower and upper bound
IN	A member of a set of specified values or a member of a subquery
NOT IN	Not a member of a set of specified values or a member of a subquery
EXISTS	'True' if a subquery returned at least one record
ANY	Compares a value to each value returned by a subquery (operator must be preceded by =, <>, >, >=, <, or <=); =Any is equivalent to In
ALL	Compares a value to each value returned by a subquery (operator must be preceded by =, <>, >, >=, <, or <=)

Examples

```
SELECT Sales_Data.Invoice_ID FROM Sales_Data
WHERE Sales_Data.Salesperson_ID = 'SP-1'
SELECT Sales_Data.Amount FROM Sales_Data WHERE Sales_Data.Invoice_ID <> 125
SELECT Sales_Data.Amount FROM Sales_Data WHERE Sales_Data.Amount > 3000
```

```
SELECT Sales_Data.Time_Sold FROM Sales_Data
WHERE Sales_Data.Time_Sold < '12:00:00'

SELECT Sales_Data.Company_Name FROM Sales_Data
WHERE Sales_Data.Company_Name LIKE '%University'

SELECT Sales_Data.Company_Name FROM Sales_Data
WHERE Sales_Data.Company_Name NOT LIKE '%University'

SELECT Sales_Data.Amount FROM Sales_Data WHERE Sales_Data.Amount IS NULL

SELECT Sales_Data.Amount FROM Sales_Data WHERE Sales_Data.Amount IS NOT
NULL

SELECT Sales_Data.Invoice_ID FROM Sales_Data
WHERE Sales_Data.Invoice_ID BETWEEN 1 AND 10

SELECT COUNT(Sales_Data.Invoice_ID) AS agg
FROM Sales_Data WHERE Sales_Data.INVOICE_ID IN (50,250,100)

SELECT COUNT(Sales_Data.Invoice_ID) AS agg
FROM Sales_Data WHERE Sales_Data.INVOICE_ID NOT IN (50,250,100)

SELECT COUNT(Sales_Data.Invoice_ID) AS agg FROM Sales_Data
WHERE Sales_Data.INVOICE_ID NOT IN (SELECT Sales_Data.Invoice_ID
FROM Sales_Data WHERE Sales_Data.Salesperson_ID = 'SP-4')

SELECT *
FROM Sales_Data WHERE EXISTS (SELECT Sales_Data.Amount
FROM Sales_Data WHERE Sales_Data.Salesperson_ID IS NOT NULL)

SELECT *
FROM Sales_Data WHERE Sales_Data.Amount = ANY (SELECT Sales_Data.Amount
FROM Sales_Data WHERE Sales_Data.Salesperson_ID = 'SP-1')

SELECT *
FROM Sales_Data WHERE Sales_Data.Amount = ALL (SELECT Sales_Data.Amount
FROM Sales_Data WHERE Sales_Data.Salesperson_ID IS NULL)
```

Logical operators

You can combine two or more conditions. The conditions must be related by AND or OR, such as:

```
salary = 40000 AND exempt = 1
```

The logical NOT operator is used to reverse the meaning, such as:

```
NOT (salary = 40000 AND exempt = 1)
```

Examples

```
SELECT * FROM Sales_Data WHERE Sales_Data.Company_Name
NOT LIKE '%University' AND Sales_Data.Amount > 3000

SELECT * FROM Sales_Data WHERE (Sales_Data.Company_Name
LIKE '%University' OR Sales_Data.Amount > 3000)
AND Sales_Data.Salesperson_ID = 'SP-1'
```

Functions

The ODBC and JDBC client drivers support many functions you can use in expressions. Some of the functions return characters strings, some return numbers, and some return dates.

Important Use uppercase letters for SQL function names (some are case sensitive).

Functions that return character strings	Description	Example
CHR	Converts an ASCII code to a one-character string	`CHR(67)` returns C
RTRIM	Removes trailing blanks from a string	`RTRIM('ABC ')` returns ABC
TRIM	Removes leading and trailing blanks from a string	`TRIM(' ABC ')` returns ABC
LTRIM	Removes leading blanks from a string	`LTRIM(' ABC')` returns ABC
UPPER	Changes each letter of a string to uppercase	`UPPER('Allen')` returns ALLEN
LOWER	Changes each letter of a string to lowercase	`LOWER('Allen')` returns allen
LEFT	Returns leftmost characters of a string	`LEFT('Mattson',3)` returns Mat
RIGHT	Returns rightmost characters of a string	`RIGHT('Mattson',4)` returns tson
SUBSTR	Returns a substring of a string, with parameters of the string, the first character to extract, and the number of characters to extract (optional)	`SUBSTR('Conrad',2,3)` returns onr `SUBSTR('Conrad',2)` returns onrad
SPACE	Generates a string of blanks	`SPACE(5)` returns ' '
STRVAL	Converts a value of any type to a character string	`STRVAL('Woltman')` returns Woltman `STRVAL(5 * 3)` returns 15 `STRVAL(4 = 5)` returns 'False' `STRVAL({12/25/2004})` returns 12/25/2004
TIME	Returns the time of day as a string	At 9:49 PM, `TIME()` returns 21:49:00
USERNAME	Returns the login ID specified at connect time	

Examples

```
SELECT CHR(67) + SPACE(1) + CHR(70) FROM Salespeople

SELECT RTRIM(' ' + Salespeople.Salesperson_ID) AS agg FROM Salespeople

SELECT TRIM(SPACE(1) + Salespeople.Salesperson_ID) AS agg FROM Salespeople

SELECT LTRIM(' ' + Salespeople.Salesperson_ID) AS agg FROM Salespeople

SELECT UPPER(Salespeople.Salesperson) AS agg FROM Salespeople

SELECT LOWER(Salespeople.Salesperson) AS agg FROM Salespeople

SELECT LEFT(Salespeople.Salesperson, 5) AS agg FROM Salespeople

SELECT RIGHT(Salespeople.Salesperson, 7) AS agg FROM Salespeople

SELECT SUBSTR(Salespeople.Salesperson_ID, 2, 2) +
SUBSTR(Salespeople.Salesperson_ID, 4, 2) AS agg FROM Salespeople

SELECT SUBSTR(Salespeople.Salesperson_ID, 2) +
SUBSTR(Salespeople.Salesperson_ID, 4) AS agg FROM Salespeople
```

```
SELECT SPACE(2) + Salespeople.Salesperson_ID AS Salesperson_ID FROM
Salespeople

SELECT STRVAL('60506') AS agg FROM Sales_Data WHERE Sales_Data.Invoice_ID
= 1

SELECT TIME() AS agg FROM Sales_Data WHERE Sales_Data.Invoice_ID = 1

SELECT USERNAME() AS agg FROM Sales_Data WHERE Sales_Data.Invoice_ID = 1
```

Functions that return numbers	Description	Example
MOD	Divides two numbers and returns the remainder of the division	`MOD(10,3)` returns 1
LEN	Returns the length of a string	`LEN('ABC')` returns 3
MONTH	Returns the month part of a date	`MONTH({01/30/2004})` returns 1
DAY	Returns the day part of a date	`DAY({01/30/2004})` returns 30
YEAR	Returns the year part of a date	`YEAR({01/30/2004})` returns 2004
MAX	Returns the larger of two numbers	`MAX(66,89)` returns 89
DAYOFWEEK	Returns the day of week (1-7) of a date expression	`DAYOFWEEK({05/01/2004})` returns 7
MIN	Returns the smaller of two numbers	`MIN(66,89)` returns 66
POW	Raises a number to a power	`POW(7,2)` returns 49
INT	Returns the integer part of a number	`INT(6.4321)` returns 6
X	Returns the decimal equivalent of a hexadecimal number	`X'b9'` returns 185
B	Returns the decimal equivalent of a binary number	`B'1001'` returns 9
ROUND	Rounds a number	`ROUND(123.456,0)` returns 123 `ROUND(123.456,2)` returns 123.46 `ROUND(123.456,-2)` returns 100
NUMVAL	Converts a character string to a number; if the character string is not a valid number, returns 0	`NUMVAL('123')` returns 123
VAL	Converts a character string to a number; if the character string is not a valid number, returns 0	`VAL('123')` returns 123

Functions that return dates	Description	Example
DATE	Returns today's date	If today is 11/21/2005, `DATE()` returns {2005-11-21}
DATEVAL	Converts a character string to a date	`DATEVAL('01/30/2006')` returns {2006-01-30}

Operator precedence

As expressions become more complex, the order in which the expressions are evaluated becomes important. This table shows the order in which the operators are evaluated. The operators in the first line are evaluated first, and so on. Operators in the same line are evaluated left to right in the expression.

Precedence	Operator
1	Unary '-', Unary '+'
2	^, **
3	*, /
4	+, -
5	=, <>, <, <=, >, >=, Like, Not Like, Is Null, Is Not Null, Between, In, Exists, Any, All
6	Not
7	AND
8	OR

The following example shows the importance of precedence:

```
WHERE salary > 40000 OR hire_date > {01/30/2004} AND dept = 'D101'
```

Because AND is evaluated first, this query retrieves employees in department D101 hired after January 30, 2004, as well as every employee making more than $40,000, no matter what department or hire date.

To force the clause to be evaluated in a different order, use parentheses to enclose the conditions to be evaluated first. For example:

```
WHERE (salary > 40000 OR hire_date > {01/30/2004}) AND dept = 'D101'
```

retrieves employees in department D101 that either make more than $40,000 or were hired after January 30, 2004.

ODBC Catalog functions

The ODBC client driver supports the following Catalog functions:

- SQLTables - catalog information is stored and reported as single part names (table name only).
- SQLColumns
- SQLColumnPrivileges
- SQLDescribeCol
- SQLGetTypeInfo

JDBC Meta Data functions

The JDBC client driver supports the following Meta Data functions:

- getColumns
- getColumnPrivileges
- getMetaData
- getTypeInfo
- getTables
- getTableTypes

Reserved SQL keywords

The following table lists reserved keywords that should not be used as names for columns, tables, aliases, or other user-defined objects. If you are getting syntax errors, these errors may be due to using one of these reserved words. If you want to use one of these keywords, you need to use quotation marks to prevent the word from being treated as a keyword.

For example, the following Create Table statement shows how to use the "OID" keyword as a data element name.

```
create table t ("oid" numeric)
```

Reserved keywords

ABSOLUTE	BREADTH	CONSTRAINT	DECLARE	EXEC
ACTION	BY	CONSTRAINTS	DEFAULT	EXECUTE
ADD	CALL	CONTINUE	DEFERRABLE	EXISTS
AFTER	CASCADE	CONVERT	DEFERRED	EXTERNAL
ALIAS	CASCADED	CORRESPONDING	DELETE	EXTRACT
ALL	CASE	COUNT	DEPTH	FALSE
ALLOCATE	CAST	CREATE	DESC	FETCH
ALTER	CATALOG	CROSS	DESCRIBE	FIRST
AND	CHAR	CURDATE	DESCRIPTOR	FLOAT
ANY	CHARACTER	CURRENT	DIAGNOSTICS	FLOOR
ARE	CHARACTER_LENGTH	CURRENT_DATE	DICTIONARY	FOR
AS	CHAR_LENGTH	CURRENT_TIME	DISCONNECT	FOREIGN
ASC	CHECK	CURRENT_TIMESTAMP	DISTINCT	FOUND
ASSERTION	CHR	CURRENT_USER	DOMAIN	FROM
ASYNC	CLOSE	CURSOR	DOUBLE	FULL
AT	COALESCE	CURTIME	DROP	GENERAL
AUTHORIZATION	COLLATE	CYCLE	EACH	GET
AVG	COLLATION	DATA	ELSE	GLOBAL
BEFORE	COLUMN	DATE	ELSEIF	GO
BEGIN	COLUMNS	DAY	END	GOTO
BETWEEN	COMMIT	DAYOFMONTH	END_EXEC	GRANT
BIT	COMPLETION	DAYOFWEEK	EQUALS	GROUP
BIT_LENGTH	CONCAT	DEALLOCATE	ESCAPE	HAVING
BOOLEAN	CONNECT	DEC	EXCEPT	HOUR
BOTH	CONNECTION	DECIMAL	EXCEPTION	IDENTIFY

Reserved keywords

IF
IFNULL
IGNORE
IMMEDIATE
IN
INDEX
INDICATOR
INITIALLY
INNER
INPUT
INSENSITIVE
INSERT
INT
INTEGER
INTERSECT
INTERVAL
INTO
IS
ISOLATION
JOIN
KEY
LANGUAGE
LAST
LCASE
LEADING
LEAVE
LEFT
LEN
LENGTH
LESS
LEVEL
LIKE
LIMIT
LOCAL
LOOP
LOWER
LTRIM
MATCH
MAX
MIN
MINUTE
MOD
MODIFY
MODULE
MONTH
NAMES
NATIONAL
NATURAL
NCHAR
NEW
NEXT
NO
NONE
NOT
NOW
NULL
NULLIF
NUMERIC
OBJECT
OCTET_LENGTH
OF
OFF
OID
OLD
ON
ONLY
OPEN
OPERATION
OPERATORS
OPTION
OR
ORDER
OTHERS
OUTER
OUTPUT
OVERLAPS
PAD
PARAMETERS
PARTIAL
PENDANT
POSITION
POW
POWER
PRECISION
PREORDER
PREPARE
PRESERVE
PRIMARY
PRIOR
PRIVATE
PRIVILEGES
PROCEDURE
PROTECTED
PUBLIC
RCASE
READ
REAL
RECURSIVE
REF
REFERENCES
REFERENCING
RELATIVE
REMOVE
REPLACE
RESIGNAL
RESTRICT
RETURN
RETURNS
REVOKE
RIGHT
ROLE
ROLLBACK
ROUND
ROUTINE
ROW
ROWS
RTRIM
SAVEPOINT
SCHEMA
SCROLL
SEARCH
SECOND
SECTION
SELECT
SENSITIVE
SEQUENCE
SESSION
SESSION_USER
SET
SIGNAL
SIMILAR
SIZE
SMALLINT
SOME
SPACE
SQL
SQLCODE
SQLERROR
SQLEXCEPTION
SQLSTATE
SQLWARNING
STATISTICS
STRUCTURE
SUBSTR
SUBSTRING
SUM
SYSTEM_USER
TABLE
TEMPORARY
TEST
THEN
THERE
TIME
TIMESTAMP
TIMEZONE_HOUR
TIMEZONE_MINUTE
TO
TRAILING
TRANSACTION
TRANSLATE
TRANSLATION
TRIGGER
TRIM
TRUE
TYPE
UCASE
UNDER
UNION
UNIQUE
UNKNOWN
UPDATE
UPPER
USAGE
USER
USERNAME
USING
VALUE
VALUES
VARCHAR
VARIABLE
VARYING
VIEW
VIRTUAL
VISIBLE
WAIT
WHEN
WHENEVER
WHERE
WHILE
WITH
WITHOUT
WORK
WRITE
YEAR
ZONE

Appendix A
Mapping FileMaker fields to ODBC data types

This table illustrates how FileMaker field types map to the standard ODBC data types.

FileMaker field type	Converts to ODBC data type	About the data type
text	SQL_VARCHAR	The maximum column length of text is 1 million characters, unless you specify a smaller Maximum number of characters for the text field in FileMaker. FileMaker returns empty strings as NULL.
number	SQL_DOUBLE	The FileMaker number field type can contain positive or negatives values as small as 10^{-308}, and as large as 10^{+308}, with up to 15 significant digits.
date	SQL_DATE	
time	SQL_TIME	The FileMaker time field type can contain the time of day or a time interval. A time interval is returned as a time of day, unless it is less than 0 or greater than 24 hours (both return a value of 0).
timestamp	SQL_TIMESTAMP	
container (BLOB)	SQL_LONGVARBINARY	You can retrieve binary data, file reference information, or data of a specific file type from a container field. Within a SELECT statement, use the CAST function to retrieve file reference information, and use the GetAs function to retrieve data of a specific file type.
calculation		The result is mapped to the corresponding ODBC data type.

String length is optional in table declarations. All strings are stored and retrieved in Unicode.

Notes

- You can SELECT up to 170 fields at one time from a FileMaker database file; you can UPDATE up to 100 fields at one time.
- FileMaker supports repeating fields (array data types), but ODBC does not. FileMaker exports repetitions to tab-delimited or comma-delimited files and separates each repetition with a group separator (Unicode decimal value 29). Text columns separated with the group separator are concatenated. All other data types return only the first repetition.

Appendix B
Mapping FileMaker fields to JDBC data types

The JDBC client driver uses the following mappings when converting FileMaker data types to JDBC SQL types. (For information about these types, see the JDK 1.5 documentation web pages at www.javasoft.com.)

FileMaker field type	Converts to JDBC SQL type
text	java.sql.Types.VARCHAR
number	java.sql.Types.DOUBLE
date	java.sql.Types.DATE
time	java.sql.Types.TIME
timestamp	java.sql.Types.TIMESTAMP
container	java.sql.Types.BLOB
calculation	specified by the data type of the calculation's result

The JDBC client driver converts the FileMaker calculation data type to the JDBC SQL type matching the calculation's result. For example, the JDBC client driver converts a FileMaker calculation that results in a timestamp data type to java.sql.Types.TIMESTAMP.

Appendix C
ODBC and JDBC error messages

Here are the basic formats of error messages you receive when working with FileMaker and ODBC/JDBC. For a listing of error numbers and explanations, see www.datadirect.com.

For more information about working with errors in FileMaker, see the Get(LastError) or Get(LastODBCError) functions described in FileMaker Pro Help.

ODBC error messages

Error messages can come from:

- ODBC driver errors
- ODBC Driver Manager errors
- SequeLink Client errors
- SequeLink Server errors
- the data source or database management system

ODBC driver error messages

An error reported by the SequeLink ODBC driver has the following format:

[DataDirect] [ODBC SequeLink driver] message

For example:

[DataDirect] [ODBC SequeLink driver] Invalid precision specified

If you get this type of error, check the last ODBC call your application made for possible problems or contact your ODBC application vendor.

ODBC Driver Manager error messages

An error reported by the ODBC Driver Manager has the following format:

[Microsoft] [ODBC Driver Manager] message

For example:

[Microsoft] [ODBC Driver Manager] Function sequence error

If you get this type of error, check to see that you have the proper ODBC support files and drivers.

SequeLink Client error messages

An error reported by the SequeLink ODBC Client has the following format:

[DataDirect] [ODBC SequeLink driver] [SequeLink Client] message

For example:

[DataDirect] [ODBC SequeLink driver] [SequeLink Client] The specified transliteration module is not found

SequeLink Server error messages

An error reported by the SequeLink Server has the following format:

[DataDirect] [ODBC SequeLink driver] [SequeLink Server] message

For example:

[DataDirect] [ODBC SequeLink driver] [SequeLink Server] Only SELECT statements are allowed in this read-only connection.

Data source error messages

An error that occurs in the data source includes the data source name, in the following format:

[DataDirect] [ODBC SequeLink driver] [data_source] message

For example, you might get the following message from your FileMaker data source:

[DataDirect] [ODBC SequeLink driver] [FileMaker] Invalid Username/Password

If you get this type of error, you did something incorrectly with the database system. Check your FileMaker documentation for more information or consult your database administrator.

Consecutive messages for errors in different columns can sometimes display an incorrect column name.

JDBC error messages

The SequeLink for JDBC driver reports errors to the calling application by returning SQLExceptions. Error messages can come from:

- JDBC driver errors
- SequeLink Server errors
- the data source or database management system

JDBC driver error messages

An error reported by the JDBC driver has the following format:

[DataDirect] [SequeLink JDBC Driver] message

For example:

[DataDirect] [SequeLink JDBC Driver] Timeout expired

If you get this type of error, check the last JDBC call your application made for possible problems or contact your JDBC application vendor.

SequeLink Server error messages

An error reported by SequeLink Server has the following format:

[DataDirect] [SequeLink JDBC Driver] [SequeLink] message

If no SequeLink Server errors exist, you see:

[DataDirect] [JDBC SequeLink driver] [SequeLink]

Data source error messages

An error that occurs in the data source includes the data source name, in the following format:

[DataDirect] [SequeLink JDBC Driver] [data_source] message

For example, you might get the following message from your FileMaker data source:

[DataDirect] [SequeLink JDBC Driver] [FileMaker] Invalid Username/Password

If you get this type of error, you did something incorrectly with the database system. Check your FileMaker documentation for more information or consult your database administrator.

Index

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y